Bayesian Linear Regression- (Parameter distribution)

Abstract—**This document contains theory behind curve fitting model**

1 Objective

Our objective is to implement the parameter distribution implemention from scratch.

2 Load Dataset

Create a sinusoidal data function

$$y = A \sin 2\pi f t + n(t) \tag{2.0.1}$$

with random noise included in the target values training set comprising N observations of t, written

$$t = \begin{pmatrix} t_1 & , & . & . & t_N \end{pmatrix}^T \tag{2.0.2}$$

together with corresponding observations of the values of y, denoted

$$y = \begin{pmatrix} y_1 & , & . & . & y_N \end{pmatrix}^T \tag{2.0.3}$$

Fig 0 was generated by choosing values of t_n, for

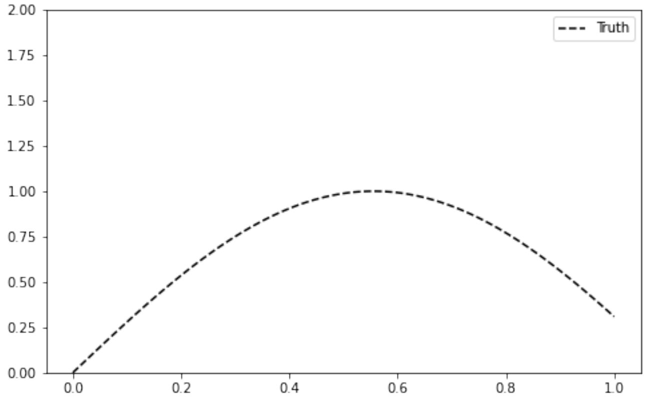

Fig. 0: Sinusoidal Dataset

n = 1, . . . , N, where N= 20 and spaced randomly. Data set y was obtained by first computing the corresponding values of the function $A \sin 2\pi f t$ and then adding a small level of random noise having a random distribution to each such point in order to obtain the corresponding value.

```
def create_data(t, noise_variance):
    return np.sin(0.9*np.pi*t) + noise(t.shape,
        noise_variance)

def noise(size, variance):
    return np.random.normal(scale=np.sqrt(
        variance), size=size)
```

From the above code we will get ϕ matrix of size (20,11) . Here column-1 will always be the value of coefficient, which will always be 1. But to create a matrix we need to consider it as a column.

$$\phi = \begin{pmatrix} 1 & t_0 & t_0^2 & \dots & t_0^{N-1} \\ 1 & t_1 & t_1^2 & \dots & t_1^{N-1} \\ 1 & t_2 & t_2^2 & \dots & t_2^{N-1} \\ \vdots & & \vdots & & \vdots \\ 1 & \dots & \dots & \dots & t_N^{N-1} \end{pmatrix} \tag{2.0.4}$$

3 Parameter distribution

For a Bayesian treatment of linear regression we need a prior probability distribution over model parameters **w**. For reasons of simplicity, we will use an isotropic Gaussian distribution over parameters **w** with zero mean.

$$p(\mathbf{w}|\alpha) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}\right) \tag{3.0.1}$$

An isotropic Gaussian distribution has a diagonal covariance matrix where all diagonal elements have the same variance α^{-1} (α is the precision of the prior. A zero mean favors smaller values of parameters $\mathbf{w_j}$ a priori. The prior is conjugate to the likelihood $p(\mathbf{t}|\mathbf{w},\beta)$ meaning that the posterior distribution has the same functional form as the prior i.e. is also a Gaussian. In this special case, the posterior has an analytical solution with the following sufficient statistics.

$$\mathbf{m}_N = \beta \mathbf{S}_N \mathbf{\Phi}^T \mathbf{t} \tag{3.0.2}$$

$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \mathbf{\Phi}^T \mathbf{\Phi} \tag{3.0.3}$$

(3.0.2) is the mean vector of the posterior and (3.0.3) the inverse covariance matrix. Hence, the posterior distribution can be written as

$$p(\mathbf{w}|\mathbf{t},\alpha,\beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N,\mathbf{S}_N) \tag{3.0.4}$$

For the moment, we assume that the values of α and β are known. Since the posterior is proportional to the product of likelihood and prior, the log of the posterior distribution is proportional to the sum of the log likelihood and the log of the prior

$$\log p(\mathbf{w}|\mathbf{t},\alpha,\beta) = -\beta E_D(\mathbf{w}) - \alpha E_W(\mathbf{w}) + \text{const.} \tag{3.0.5}$$

```
def posterior(Phi, t, alpha, beta, return_inverse=
    False):
    S_N_inv = alpha * np.eye(Phi.shape[1]) +
        beta * Phi.T.dot(Phi)
    S_N = np.linalg.inv(S_N_inv)
    m_N = beta * S_N.dot(Phi.T).dot(t)
    if return_inverse:
        return m_N, S_N, S_N_inv
    else:
        return m_N, S_N
```

Plot the posterior and actual data.

```
def plot_data(x, t):
    plt.scatter(x, t, marker='o', c="k", s=20)


def plot_truth(x, y, label='Truth'):
    plt.plot(x, y, 'k--', label=label)


def plot_posterior_samples(x, ys, plot_xy_labels
    =True):
    plt.plot(x, ys[:, 0], 'r-', alpha=0.5, label='
        Post. samples')
    for i in range(1, ys.shape[1]):
        plt.plot(x, ys[:, i], 'r-', alpha=0.5)

    if plot_xy_labels:
        plt.xlabel('x')
        plt.ylabel('y')
```

Download Python codes from

```
https://github.com/ayushkesh/EE4015/tree/master/
    A5
```

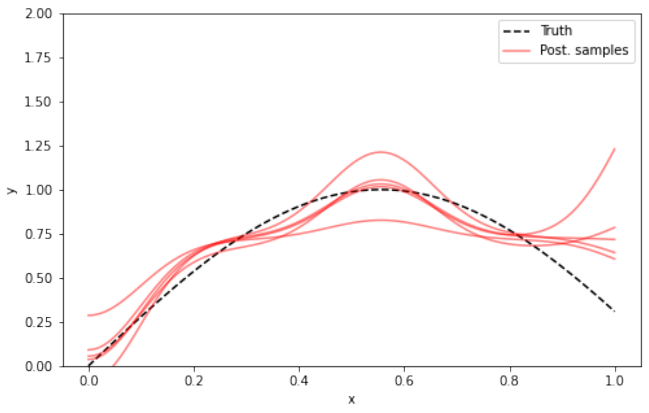

Fig. 0: Sinusoidal Dataset